Departamento de Engenharia de Computação e Automação Industrial

Faculdade de Engenharia Elétrica e de Computação

Universidade Estadual de Campinas

RELATÓRIO DE PROJETO DE PESQUISA

DE PÓS-DOUTORADO

(Proc. No. 2008/51041-0):

Interação Independente da Aplicação Com Dados Volumétricos Processados na GPU

PROPONENTE: HARLEN COSTA BATAGELO

SUPERVISORA: WU, SHIN - TING

Abril de 2009

Resumo

Neste relatório apresentamos os resultados do projeto de pós-doutorado no qual exploramos a possibilidade de estender a arquitetura de interação 3D proposta na tese de doutorado de modo a desenvolver mecanismos eficientes e acurados de interação com dados volumétricos médicos processados na GPU. Ao longo do desenvolvimento do projeto, as atividades se concentraram no uso de imagens neurológicas 3D obtidas de equipamentos de ressonância magnética do Hospital das Clínicas da Unicamp, com apoio do LNI (Laboratório de Neuroimagem) da Faculdade de Ciências Médicas da mesma universidade. Dentro das premissas estabelecidas, os objetivos propostos no projeto foram alcançados de forma satisfatória, a saber a extensão da arquitetura de interação para interação com dados volumétricos visualizados como isosuperfícies. Resultados preliminares dessa extensão despertaram a atenção dos neurologistas e neurocirurgiões do LNI quanto à possibilidade de utilizar ferramentas de interação 3D para facilitar o diagnóstico de anomalias com extensões superficiais maiores do que sua espessura, tal como lesões displásticas do córtex, bem como oferecer suporte rápido, não-invasivo e sem custos ao planejamento cirúrgico. Neste relatório detalhamos o desenvolvimento de uma dessas ferramentas: a *reformatação curvilinear* usando manipulação direta 3D. Apresentamos também os resultados preliminares obtidos pelo uso dessa ferramenta com dados reais de diferentes tipos de lesões, e sugestões de trabalhos futuros que objetivam corrigir limitações ainda existentes para seu uso em ambiente clínico. Em especial, verificamos que a não-homogeneidade das densidades obtidas nos dados de ressonância magnética do interior do cérebro dificultam a definição de isosuperfícies suaves nessa região. Tais imagens são preferencialmente visualizadas através da técnica de renderização volumétrica direta, permitindo ajustes de parâmetros gráficos de forma que os dados reais não necessariamente condizem com os dados percebidos pelo especialista. Desse modo, o cálculo de atributos geométricos para interação deve levar em conta essas modificações.

1 Introdução

O atual paradigma de interação e sua dependência da existência de modelos geométricos na memória do sistema em oposição ao uso de atributos geométricos locais disponibilizados pela GPU tem sido questionado desde 1992 pelo nosso grupo de pesquisa coordenado pela Prof[a]. Wu Shin-Ting [11]. Tal paradigma, utilizado desde o surgimento da interação por manipulação direta [20] e apropriado para a tecnologia que reinou até o final do século XX, supõe que o modelo visualizado não tem sua aparência modificada significativamente no processamento ocorrido no fluxo de visualização, de modo que todo procedimento de interação pode ser calculado exclusivamente com relação a esse modelo original. Hoje em dia, entretanto, há muitos artifícios utilizados no fluxo de visualização para melhorar a eficiência ou qualidade da representação visual dos modelos, tais como procedimentos de filtragem e texturização. Esses procedimentos, dependendo do contexto podem variar de tal forma que o que se vê não é mais necessariamente parecido com o que está representado.

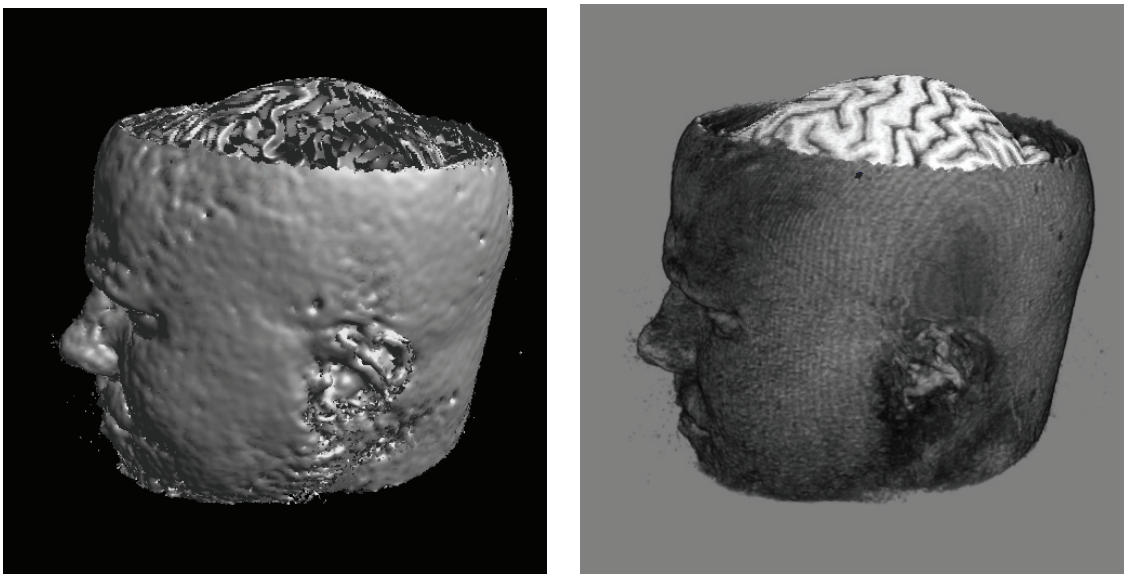
Motivado pelo trabalho seminal de Wu *et al.* [23], foi proposta uma arquitetura de interação no trabalho do meu doutorado tendo como objetivo fazer uma mediação entre os modelos originais e os modelos percebidos pelo usuário por meio de propriedades geométricas diferenciais, de forma que a lacuna entre o que se vê e o que se percebe seja a menor possível e assim assegure que o usuário tenha a real sensação do que está manipulando. No caso em que as modificações da geometria do fluxo de visualização são ditadas pelo aplicativo (por exemplo, em um algoritmo de deslocamento de vértices), as funções de transformação podem ser programadas a priori na GPU, e a arquitetura de interação se encarrega de estimar os atributos de geometria diferencial necessários à manipulação direta. Outras distorções geométricas podem ocorrer apenas no processador de fragmentos. Neste caso, a arquitetura assume que o *shader* de fragmentos provê as propriedades geométricas locais em cada amostra, tal como ocorre com um algoritmo de *relief mapping* [14]. Neste trabalho de pós-doutorado exploramos o uso dessa arquitetura de interação para a manipulação de dados volumétricos. Neste

caso, o procedimento de visualização concentra-se também no processador de fragmentos.

O desenvolvimento do projeto contou com 3 fases distintas. Nos primeiros quatro meses de implementação nos preocupamos com a leitura dos arquivos de imagens médicas disponibilizados pelos equipamentos de aquisição. Nesta etapa entramos em contato com os profissionais do Laboratório de Neuroimagem (LNI) do Hospital das Clínicas da Unicamp. O LNI trabalha especialmente com diagnóstico de epilepsia utilizando um equipamento de aquisição de imagens por ressonância magnética. Considerando o interesse demonstrado pelos médicos do LNI na possível aplicação dos resultados do projeto em seu laboratório, aliado à nossa facilidade de entrar em contato com esses profissionais e facilidade de obter dados clínicos com rapidez, optamos por estabelecer um vínculo com essa equipe durante todo o desenvolvimento do trabalho de pós-doutorado. Assim, tivemos à nossa disposição uma grande quantidade de dados neurológicos, seja de pacientes sem lesões, com lesões, e imagens de pré e pós-operatório. Contamos ainda com o suporte presencial dos médicos para dirimir dúvidas sobre o formato dos dados e sobre as ferramentas de interação desejadas. Essa interação com uma equipe de médicos local trouxe mais agilidade na etapa de análise de requisitos das ferramentas.

Em uma segunda fase do projeto, após implementar a leitura de arquivos de imagens médicas do LNI, nos dedicamos a estudar a configuração da arquitetura de interação apresentada no trabalho de doutorado de modo a trabalhar com dados volumétricos. Aproveitamos a arquitetura de Stegmaier *et al.* para visualização de dados volumétricos usando *ray-casting* na GPU [19]. Nessa arquitetura, os dados volumétricos são armazenados como texturas 3D: uma para os valores de densidade e outra para os gradientes. No fluxo de visualização, um *shader* de fragmentos amostra as texturas 3D de modo a visualizar o modelo como *isosuperfícies* ou através de *renderização volumétrica direta*.

No modo de visualização de isosuperfícies, exibem-se superfícies que representam pontos de densidade constante do volume, de acordo com um *isovalor* que corresponde a tal



(a) (b)

Figura 1: Modo de visualização de (a) isosuperfícies e (b) renderização volumétrica direta do modelo de uma cabeça recortada na parte superior através de reformatação curvilinear.

densidade. A superfície exibida é sempre opaca e sombreada de acordo com o gradiente correspondente à posição dos pontos da superfície (figura 1.a).

No modo de *renderização volumétrica direta*, os valores de densidade do volume são mapeados para valores de opacidade e cor segundo uma função de transferência, e combinados ao longo do raio do *ray-caster* de acordo com a equação de renderização (figura 1.b) [5].

Na figura 1, chamamos atenção para a diferença de apresentação dos resultados usando visualização de isosuperfícies e renderização volumétrica direta. No modo de visualização de isosuperfícies, a pele é percebida como uma superfície relativamente suave, mas o interior do cérebro é exibido de forma ruidosa em razão das sutis diferenças de densidade existentes nessa região, e que por isso correspondem a diferentes isosuperfícies. Por outro lado, no modo de renderização volumétrica direta, a visualização dos sulcos cerebrais é relativamente suave uma vez que a combinação de valores de opacidade ditada pela função de transferência

produz um efeito de suavização.

O procedimento de *ray-casting* utilizado é muito semelhante ao empregado por algoritmos de mapeamento de detalhes 3D baseados em *ray-casting*, tal como o algoritmo de *relief mapping* utilizado no trabalho de doutorado. Assim, apenas com uma alteração da configuração da arquitetura de interação, conseguimos utilizá-la para a interação com isosuperfícies calculadas por *ray-casting*.

De maneira concomitante com a realização da segunda fase do projeto, a terceira fase contou com a implementação de uma ferramenta de manipulação volumétrica direta 3D de modo a validar a configuração da arquitetura de interação com isosuperfícies. Em especial, desenvolvemos em conjunto com os médicos no LNI, uma ferramenta de *reformatação curvilinear* utilizada no diagnóstico de displasias. Assim, as contribuições principais deste trabalho de pós-doutorado foram:

- Configuração da arquitetura de interação 3D de Batagelo e Wu [2] de modo a suportar dados volumétricos visualizados como isosuperfícies.
- Desenvolvimento de uma ferramenta de reformatação curvilinear usando manipulação volumétrica direta 3D, com as seguintes vantagens sobre as técnicas atuais: (1) Utilização de uma metáfora de pintura com pincel de modo a facilitar o trabalho do usuário na segmentação da superfície do cérebro; (2) Possibilidade de trabalhar diretamente no espaço nativo da aquisição dos dados, preservando distâncias e ângulos e possibilitando assim seu uso em planejamento cirúrgico; (3) Reformatação curvilinear em tempo de interação, de modo que o procedimento de pintura, segmentação e reformatação pode ser realizado em segundos em vez de minutos, como ocorre nas técnicas atuais [3, 8].

Verificamos que a natureza dos dados de ressonância magnética e a técnica utilizada para computar isosuperfícies não são capazes de prover uma visualização de superfícies suaves, como seria possível com a técnica de renderização volumétrica direta, para manipulação de características do interior do cérebro tais como sulcos e giros. Para a ferramenta atual

de reformatação curvilinear isso não compromete a qualidade de auxílio ao diagnóstico, uma vez que a pintura 3D é realizada em cima do escalpo, o qual apresenta gradientes mais homogêneos, cuja visualização é bem similar à da renderização volumétrica direta e corresponde a uma superfície suave no modo de renderização de isosuperfícies. Assim, em nosso protótipo o usuário visualiza o modelo no modo de renderização volumétrica direta, mas a interação é realizada internamente segundo o modo de renderização por isosuperfícies.

Em nossas reuniões com os especialistas, observamos que a renderização volumétrica direta, por ser mais flexível na configuração e por proporcionar uma imagem esteticamente mais agradável, é a preferida. Portanto, é desejável que seja possível interagir com as características do interior do cérebro através das imagens geradas pela renderização volumétrica direta. Assim como na visualização, o cálculo dos atributos geométricos diferenciais para interação deve levar em consideração a avaliação da função de transferência e integração da equação de renderização, tornando o problema mais desafiador. Essa tarefa, necessária para que o usuário sinta como se estivesse interagindo com aquilo que ele vê, é o foco de nossos trabalhos futuros. Conjeturamos que podemos realizar tais interações ao definir superfícies como interfaces de meios obtidos da variação dos valores da função de transferência.

2 Desenvolvimento do Projeto

Nesta seção detalhamos o desenvolvimento deste projeto segundo as três etapas citadas na seção 1. A primeira, na qual entramos em contato com os médicos e tivemos acesso aos dados de ressonância magnética, compreende a implementação de um leitor de arquivos de imagens médicas. A segunda, e principal etapa do projeto, compreende a configuração da arquitetura de interação 3D de Batagelo e Wu [2] de modo a trabalhar com dados volumétricos. Para validar a configuração realizada (terceira etapa), detalhamos a implementação de uma ferramenta de reformatação curvilinear usando essa arquitetura.

2.1 Leitura de Arquivos de Imagens Médicas

Como primeira etapa do desenvolvimento deste projeto, realizamos a implementação de ferramentas de leitura dos arquivos de dados gerados pelos equipamentos de aquisição de imagens médicas. Em um contato inicial com o o Prof. Carlos Arturo Levi D'Ancona, médico urologista de Campinas-SP, tivemos acesso a imagens de tomografia computadorizada gravadas em formato DICOM (*Digital Imaging and Communications in Medicine*) [12]. Tal formato é considerado padrão de indústria na área de equipamentos médicos de aquisição de imagens. Utilizamos então a biblioteca GDCM (*Grassroots DICOM library*) [10] para facilitar a tarefa de leitura do conteúdo dos arquivos. Uma vez criado o leitor de arquivos DICOM e o protótipo de um visualizador simples, entramos em contato com os profissionais no LNI, por sugestão do Prof. D'Ancona ao ressaltar que as imagens neurológicas são muito mais complexas que as imagens renais. A partir de então, nos concentramos na leitura dos formatos de arquivos gerados pela máquina de ressonância magnética utilizada no Hospital das Clínicas da Unicamp. Neste momento, o hospital conta com um equipamento Elscint Prestige 2T que gera arquivos em formato DICOM. Descobrimos, entretanto, que tal máquina emprega um rótulo diferente para indicar o arranjo de *bytes* que compõe cada imagem obtida pelo aparelho. Este rótulo não segue a especificação DICOM e, segundo informações obtidas dos desenvolvedores do GDCM, consiste na identificação de um arranjo de dados compactados segundo o algoritmo de compactação RICE [16] (modo de compactação sem perdas baseado nos códigos de compactação de Golomb [7]), porém com modificações não documentadas. Informações sobre acesso a esses dados não são mais fornecidas pela Elscint, uma vez que esta empresa, após ter sido adquirida pela General Electric, deixou de oferecer suporte aos desenvolvedores. Após tentativas mal-sucedidas de descompactar tais dados, chegamos à conclusão, em conjunto com os desenvolvedores da biblioteca GDCM, que um processo de engenharia reversa seria necessário para desenvolver um leitor desses arquivos. Uma vez que tal trabalho seria muito dispendioso, resolvemos abandonar o uso desse formato DICOM

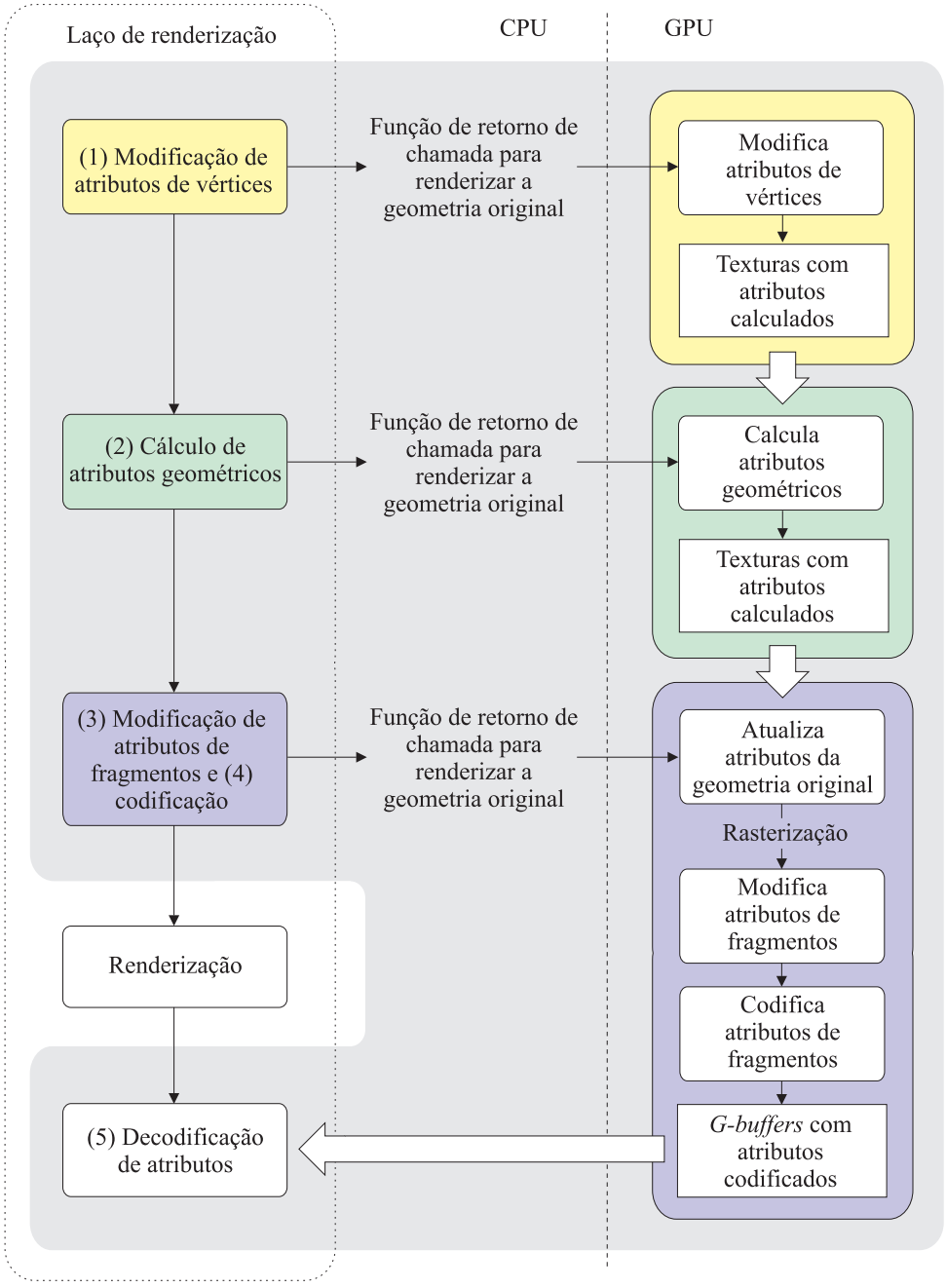


Figura 2: Estágios de processamento da arquitetura de interação utilizando malhas triangulares. O fluxo de dados de textura é indicado pelas setas grossas.

modificado. Por outro lado, verificamos que o LNI dispõe de um servidor dedicado de imagens Silicon Graphics que executa um sistema desenvolvido pela Elscint capaz de ler tal formato e exportar em um formato DICOM descompactado. Apesar de tal *software* ser obsoleto, assim como o equipamento de aquisição, optamos por desenvolver um leitor desse formato de arquivo, o que foi feito rapidamente graças ao uso do GDCM. Felizmente, no momento em que este relatório está sendo escrito, o Hospital das Clínicas da Unicamp está em processo de instalação de um novo equipamento de ressonância magnética capaz de gerar arquivos DICOM compatíveis com o GDCM, sem necessidade de utilizar *softwares* legados de conversão.

Desenvolvemos também um conversor dos arquivos DICOM para o formato de arquivo RAW utilizado pelo visualizador OpenQVis [15] e pelo protótipo de visualização de dados volumétricos através de *ray-casting* na GPU desenvolvido por Stegmaier *et al.* [19]. Tal visualizador através de *ray-casting* foi utilizado como base para desenvolvermos nosso protótipo.

2.2 Configuração da Arquitetura de Interação 3D

A arquitetura de interação que motivou a realização deste trabalho tem como característica principal a capacidade de prover, em uma escala de *pixel* a *pixel*, dados definidos pela aplicação e atributos de geometria diferencial discreta de modelos processados em um fluxo programável de renderização [2]. Esta arquitetura é capaz de processar geometria existente apenas na memória de vídeo, e é centralizada na suposição de que o usuário espera interagir com aquilo que ele vê. O fluxo de visualização de dados volumétricos da arquitetura de Stegmaier *et al.* é compatível com este paradigma, no sentido de que os dados 3D são armazenados como dados de textura 3D processadas pela GPU utilizando uma técnica de *ray-casting* [19]. Isso nos levou a integrar ambas as propostas de modo a implementar as tarefas de manipulação direta necessárias à ferramenta que decidimos implementar em conjunto com os profissionais do LNI: a ferramenta de reformatação curvilinear. Como mostramos

no documento de projeto de pós-doutorado, nossa arquitetura de interação foi originalmente projetada de modo a tratar apenas superfícies descritas por malhas triangulares. Nesta seção mostramos como configuramos a arquitetura de modo a considerar dados volumétricos visualizados pela arquitetura de *ray-casting* na GPU.

Um esquema simplificado dos estágios de processamento da arquitetura de interação é mostrado na figura 2. A integração desta arquitetura (região sombreada em cinza claro) com o laço de renderização da aplicação, e as configurações propostas para tratar dados volumétricos neste fluxo de processamento são descritas a seguir:

1. **Modificação de atributos de vértices**: Este estágio é executado no processador de vértices quando a arquitetura chama uma função de chamada de retorno (*callback*) contendo as funções de desenho das primitivas que sofrem mudanças de atributos de vértices. O *shader* de vértices neste caso deve ser uma função, definida pela aplicação, que modifica os atributos dos vértices. Tais atributos são armazenados em texturas para processamento adicional em etapas ulteriores da arquitetura.

   Considerando os dados volumétricos visualizados através de *ray-casting*, o processamento ocorre quase que somente no nível dos fragmentos, uma vez que as primitivas enviadas ao fluxo de visualização não são alteradas. Em especial, tais primitivas são apenas faces da caixa limitante dos dados volumétricos, utilizadas para determinar a posição inicial e trajetória de cada raio do algoritmo de *ray-casting*. Sendo assim, não precisamos levar em consideração as modificações dos atributos de vértices ao tratar de dados volumétricos desenhados por *ray-casting*.

2. **Cálculo dos atributos geométricos**: Esta etapa é executada no processador de vértices, novamente quando a arquitetura chama a função de chamada de retorno com a função de desenho de cada modelo. Atributos de geometria diferencial são estimados para cada vértice da malha triangular, com base nos dados obtidos das texturas de saída do estágio anterior. Os resultados (atributos de geometria diferencial discreta)

são gravados em novas texturas para uso posterior.

Para dados volumétricos visualizados com *ray-casting*, o cálculo de atributos de geometria diferencial com base nos vértices não é necessário, uma vez que os dados necessários são atributos geométricos calculados em amostras de *voxels*. Atributos de geometria diferencial devem então ser estimados no *shader* de fragmentos, quando os dados volumétricos são amostrados para visualização. Ao utilizarmos a arquitetura de visualização com *ray-casting*, já temos à nossa disposição os gradientes pré-computados na CPU para cada *voxel* e armazenados em uma textura 3D na GPU. Dessa maneira, não precisamos calcular tais atributos em tempo real. Na GPU, os gradientes podem ser utilizados para calcular propriedades tais como as curvaturas e direções principais de pontos de isosuperfícies. Entretanto, a estimativa desses atributos não foi implementada até este momento pois a ferramenta de interação implementada não necessitou de tais atributos.

3. **Modificação dos atributos de fragmentos**: Este estágio tem início em um novo passo de renderização, utilizando um *shader* de vértices que amostra as texturas de saída utilizadas nos estágios anteriores, e atualiza os atributos dos vértices de acordo com as modificações ocorridas nesses estágios, imediatamente antes da rasterização. Este estágio também é disparado por uma função de chamada de retorno definida na aplicação. Durante a rasterização, tais atributos são interpolados para cada fragmento. Uma função definida pela aplicação, tendo como entrada os atributos de fragmento, é executada de modo a modificar tais atributos.

   Considerando dados volumétricos visualizados com *ray-casting* na GPU, tal estágio executa os algoritmos de recorte (*cropping*) que controlam o descarte de *voxels* antes do cálculo de atributos geométricos. Por exemplo, na técnica de reformatação curvilinear que implementamos, a operação de remover camadas do cérebro é efetivada aqui, ao excluir da integração da equação de visualização os *voxels* que foram marcados para

remoção.

4. **Codificação dos atributos de fragmentos**: Este estágio é realizado imediatamente após a execução do estágio anterior, porém no mesmo *shader* de fragmentos. Os atributos modificados dos fragmentos são codificados como componentes de cor em *buffers* de saída não-visíveis, em formato de ponto flutuante. Este estágio não possui nenhuma modificação quando consideramos dados volumétricos.

5. **Decodificação dos atributos de fragmentos**: Sob demanda, os *buffers* de saída do estágio anterior são transferidos para a memória do sistema. Os atributos são então decodificados e ficam disponíveis para uso da aplicação.

   Tal procedimento não é alterado quando consideramos dados volumétricos, uma vez que os dados resultantes em ambos os casos (malhas triangulares e dados volumétricos) estão agora disponíveis apenas no domínio do espaço da imagem. Cada *pixel* da imagem contém os atributos de geometria diferencial discreta e os dados definidos pela aplicação, os quais podem ser lidos e utilizados arbitrariamente para realização das tarefas de manipulação direta.

3 Reformatação Curvilinear

Para validar a configuração da arquitetura de interação utilizando dados volumétricos, desenvolvemos ferramentas de reformatação curvilinear e recorte irregular (*irregular cropping*) capazes de dar ao médico suporte ao diagnóstico de displasias corticais focais.

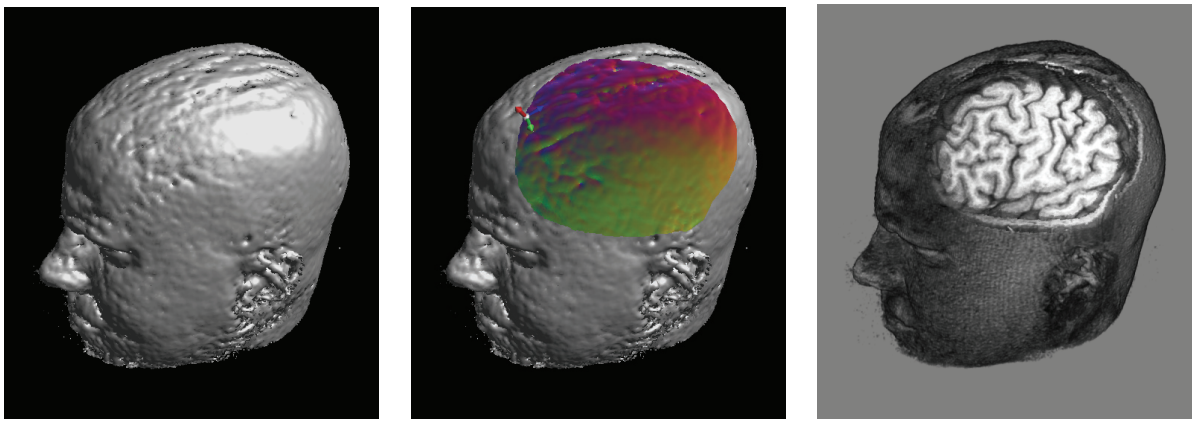
Displasia cortical focal é uma causa de epilepsia refratária (*i.e.*, que resiste a tratamentos medicamentosos). Geralmente, o tratamento indicado é a remoção cirúrgica da área lesionada, com elevadas taxas de sucesso. Embora essas lesões sejam cada vez mais facilmente detectadas através das imagens de alta qualidade fornecidas por equipamentos de ressonância magnética, resultando também em um melhor controle do procedimento cirúrgico a ser ado-

tado nos pacientes tratados, pequenas lesões de displasia ainda podem ser ignoradas. Isso ocorre porque planos de corte de imagem convencionais não são suficientes para revelar todas as sutis anormalidades estruturais que podem ocorrer. Bastos *et al.* mostraram que a detecção de lesões displásticas focais sutis pode ser obtida quando se melhora a visualização anatômica da estrutura dos giros do cérebro através de reformatação curvilinear [1]. De fato, a reformatação curvilinear de dados tridimensionais obtidos por ressonância magnética é reconhecida pela comunidade médica como uma das mais úteis ferramentas não-invasivas de exibição da estrutura cerebral. Ela consiste em tornar possível a visualização de fatias curvilineares concêntricas do cérebro, eqüidistantes entre si e em profundidades progressivas a partir de uma superfície de referência.

Há dois modos de realizar a tarefa de reformatação curvilinear usando uma superfície de referência: criando tal superfície através do delineamento manual de curvas-guias em várias fatias bidimensionais, ou realizando uma segmentação automática do cérebro de modo a extrair essa superfície. Na primeira abordagem, implementada no *software* BrainSight [17] utilizado no LNI, os neurologistas têm mais controle sobre o que eles vão obter, mas o procedimento é cansativo e consome muito tempo. Na segunda abordagem, empregada por Hupertz *et al.* [8], a segmentação correta depende do ajuste dos dados volumétricos para um espaço normalizado no qual a reformatação é aplicada. Essa normalização, necessária em razão das pequenas diferenças entre formatos de cérebros dos pacientes, é computacionalmente custosa e faz com que a reformatação não seja mais realizada em espaço nativo, isto é, há distorção de distâncias e ângulos. Uma transformação inversa ao espaço nativo é possível, porém a custos computacionais adicionais.

De modo a contornar as dificuldades existentes em ambos os métodos, desenvolvemos uma ferramenta capaz de realizar a segmentação de uma superfície através de uma metáfora de pintura em 3D. A arquitetura de interação foi utilizada para implementar uma ferramenta que permitisse a segmentação manual do cérebro através de movimentos simples do cursor nos

dados volumétricos visualizados. Para a visualização dos dados volumétricos, nos baseamos na arquitetura de *ray-casting* na GPU proposta por Stegmaier *et al.* [19], e cujo código-fonte encontra-se disponível em [18].



(a) (b) (c)

Figura 3: Reformatação curvilinear utilizando manipulação direta: (a) imagem 3D visualizada como isosuperfícies usando *ray-casting*; (b) superfície de referência, colorida de acordo com o valor do vetor normal estimado da malha; e (c) visualização da reformatação curvilinear em uma profundidade de 35 *voxels*, usando renderização volumétrica direta.

A figura 3.a mostra uma imagem volumétrica 3D de ressonância magnética e a aplicação de nossa técnica de reformatação curvilinear. Após posicionar o cursor sobre uma isosuperfície visível e alterar o estado da aplicação para um modo de pintura, o usuário pressiona o botão esquerdo do *mouse* e realiza a operação de pintura do escalpo. A posição 3D da isosuperfície correspondente ao escalpo em cada *pixel* desenhado é utilizada para gerar uma malha triangular, a qual será a superfície de referência da reformatação curvilinear. Na figura 3.b, tal malha é desenhada com cores de acordo com o vetor normal estimado. A malha triangular é então voxelizada de modo a calcular os *voxels* que a intersectam. Tais *voxels* são então removidos, dando a impressão que o volume foi "descascado". Através do deslocamento sucessivo dessa malha triangular ao longo do vetor normal em cada vértice,

superfícies discretas cuja distância entre si é de pelo menos 1 *voxel* são criadas em profundidades cada vez maiores, formando conjuntos de superfícies eqüidistantes. Ao voxelizar e remover os *voxels* correspondentes dessas superfícies, o usuário consegue visualizar o interior do volume. A figura 3.c ilustra o volume após remoção de uma série de superfícies discretas que totalizam uma profundidade de aproximadamente 35 *voxels*.

A seguir detalhamos a implementação da técnica proposta.

3.1 Segmentação Usando Manipulação Direta

Em vez de utilizar a visualização usual de planos de imagem, a saber os planos sagital, coronal e axial, a reformatação curvilinear permite visualizar uma série de superfícies de deslocamento a partir de uma superfície de referência, visualizadas em 3D ou mapeadas em 2D. Um passo fundamental para realizar a reformatação curvilinear é ajustar corretamente a superfície de referência de modo que esta se situe rente à superfície do cérebro, imediatamente sobre os sulcos cerebrais, ou de tal modo a seguir a curvatura do cérebro. Assim, tal superfície pode ser obtida também da superfície do crânio, ou do escalpo. Uma forma possível de obter essa superfície consiste em segmentar o cérebro, crânio ou escalpo. Há duas abordagens para obter isso: usando segmentação automática [3, 8] ou segmentação guiada pelo usuário [17].

Embora a superfície externa do cérebro seja claramente distinguível nas imagens de ressonância magnéticas, sua segmentação automática é um problema difícil e de alto custo computacional [4]. A partir de nossas observações e entrevistas com os profissionais do LNI, verificamos que uma segmentação guiada por um especialista é ainda considerada mais confiável e acurada se a visualização for apropriada. As ferramentas atuais de segmentação manual são, entretanto, limitadas a manipulações bidimensionais. As curvas de direção das superfícies devem ser desenhadas manualmente em uma série de fatias de um dado plano de imagem (sagital, coronal e axial). Não há regras definidas sobre o espaçamento das curvas de direção, mas quanto mais curvas houverem, melhor será a representação da superfície 3D

de referência resultante. A principal reclamação dos usuários é que esse procedimento requer muito tempo e é suscetível a erros resultantes da fadiga. Os médicos entrevistados relataram que, por esses motivos, ferramentas que dependem dessa técnica acabam sendo desprezadas no ambiente de rotina clínica [17].

Entre as características mais desejadas pelos neurologistas em um sistema de interação com imagens médicas, a mais relevante é a capacidade de interagir em 3D com imagens de alta resolução. Em especial, é desejada uma ferramenta de interação capaz de fornecer uma realimentação visual que proporcione a impressão de que o médico está dissecando livremente camadas do cérebro. À esta interação deve corresponder uma visualização em tempo real das informações contidas no interior do cérebro, ajudando assim o médico a esclarecer as dúvidas sobre as hipóteses de diagnóstico. Enfim, é desejável interagir como se o cérebro estivesse nas mãos do médico, mas ao mesmo tempo com ações que causem o mínimo de fadiga.

Seria útil permitir que um usuário especialista pudesse guiar a segmentação através de poucas (e simples) ações de interação. Assim, propomos uma solução usando uma ferramenta de desenho com a qual o usuário pode desenhar livremente áreas de interesse sobre o modelo visualizado, utilizando um pincel virtual. A região pintada é utilizada como uma segmentação do escalpo. Essa segmentação é suficiente pois ela segue a curvatura do crânio, que é o desejado para realizar a reformatação curvilinear.

Em essência, queremos mapear um conjunto de *pixels* selecionados através de pintura pelo dispositivo apontador 2D em um conjunto de *voxels* que correspondem a um envelope do cérebro. Propomos para isso empregar uma malha triangular como objeto geométrico intermediário. Essa malha triangular serve não apenas para melhor aproximar a camada exterior do cérebro (ou o escalpo), mas também para derivar de forma mais acurada as superfícies de deslocamento utilizadas na reformatação.

3.2 Malhas Triangulares como Superfícies de Referência

É conceitualmente simples determinar o *voxel* correspondente quando o usuário pinta um *pixel*, uma vez que essa informação pode ser disponibilizada facilmente através da arquitetura de interação (seção 2). Em uma etapa preliminar do projeto, realizamos a reformatação curvilinear através da simples remoção desses *voxels* e propagação deles para *voxels* das camadas mais profundas. Por outro lado, verificamos que após selecionar a região de interesse, é natural que o usuário queira visualizar tal região em diferentes pontos de vista e níveis de aproximação [22]. Para visualizar em um nível maior de aproximação, certamente mais *pixels* devem ser utilizados na visualização da mesma região de interesse. Se isso não ocorrer, o usuário visualizará a região como uma grade de pontos desconexos. Para evitar esse efeito indesejável de descontinuidade visual, lemos as coordenadas 3D da isosuperfície disponibilizadas pela arquitetura em cada amostra de *pixel* e utilizamos tais coordenadas como coordenadas de vértices de uma malha triangular. Tal malha triangular é então desenhada de modo a garantir a continuidade visual das amostras. Também utilizamos tal malha em um processo de voxelização no qual rotulamos os *voxels* que devem ser removidos. Isso mantém a continuidade da camada de *voxels* a ser removida.

Exploramos a regularidade do arranjo de *pixels* no espaço da tela de modo a construir a malha. Introduzimos um vértice na posição 3D correspondente de cada *pixel* (x_i, y_j) e conectamos triplas $[(x_i, y_j), (x_i, y_{j+1}), (x_{i+1}, y_{j+1})]$ ou $[(x_i, y_j), (x_{i+1}, y_{j+1}), (x_{i+1}, y_j)]$ para formar faixas de triângulos, como ilustrado na figura 4. Se um vértice de uma tripla não existe, o triângulo é simplesmente descartado, como ocorre com o vértice $\mathbf{v}$ na figura.

3.3 Suavização da Malha

Nos modelos utilizados em nossos testes, verificamos que as isosuperfícies do escalpo podem conter buracos que não correspondem ao que é esperado pelo usuário. Isso ocorre porque o escalpo não corresponde a apenas um valor de densidade em toda a extensão de sua superfície.

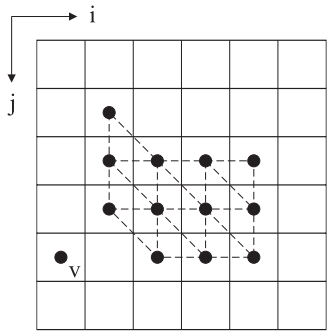


Figura 4: Malha triangular com conectividade definida segundo regularidade do espaço da imagem.

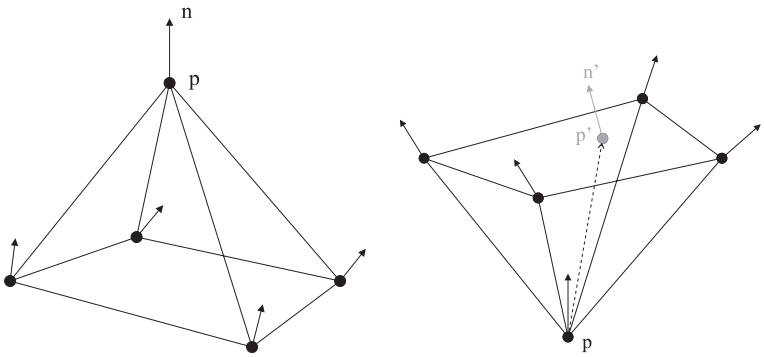
Assim, algumas amostras da pintura não necessariamente contribuem para a formação de uma malha triangular que toca apenas o escalpo.

Nossa solução para corrigir tal problema consiste em suavizar a malha triangular através do deslocamento de amostras na direção do plano que toca os máximos locais. O algoritmo utiliza uma atualização recursiva da posição $\mathbf{p}$ e vetor normal $\mathbf{n}$ de cada amostra de acordo com o plano que aproxima suas m amostras adjacentes $\mathbf{v}_i$ até que todos os mínimos locais tenham sido removidos. Utilizamos o método de Newell para estimar a equação do plano $n_x x + n_y y + n_z z + d = 0$ para uma seqüência de vértices adjacentes [21]. Se o plano está abaixo de $\mathbf{p}$, como mostra a figura 5.a, a amostra é um máximo local em potencial. Neste caso, nada é feito. Ao contrário, se o plano está acima de $\mathbf{p}$, como mostra a figura 5.b, a posição e o vetor normal da amostra são, respectivamente, substituídos por

$$\begin{aligned} \mathbf{p}' &= \frac{\sum_{i=1}^{n} \mathbf{v}_i}{n} \\ \mathbf{n}' &= \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}. \end{aligned}$$

Os novos vetores são mostrados em cinza. Em nossa implementação, realizamos esse procedimento de forma iterativa, até que todos os pequenos buracos da isosuperfície do escalpo

fossem removidos. Para os dados por nós utilizados, o número de iterações que utilizamos foi 70, estimado empiricamente.



(a) (b)

Figura 5: Procedimento de suavização da malha.

3.4 Deslocamento da Malha

Uma vez que obtemos a malha triangular suavizada, esta deve ser deslocada em uma dada direção de modo a formar camadas eqüidistantes entre si. Experimentamos diferentes direções de deslocamento da malha de modo a verificar suas vantagens de implementação e resultados visuais obtidos.

Inicialmente, deslocamos cada vértice da malha triangular na direção determinada entre o vértice atual e o centróide do volume 3D. Desse modo, cada deslocamento corresponde a uma malha triangular de área menor que a anterior, até que a malha se reduza a um ponto ao atingir o centróide do volume (aproximadamente o centro da cabeça). Uma vez que a superfície de referência não possui auto-oclusões (isso porque a superfície é formada apenas a partir de *pixels* visíveis na pintura 3D) as malhas deslocadas não sofrem auto-interseções quando cada vértice é deslocado em direção ao centróide. De fato, a auto-interseção só ocorre neste ponto, cujo tratamento é trivial pois basta determinar o *voxel* correspondente ao ponto. A desvantagem deste procedimento é que o deslocamento não segue a curvatura da superfície

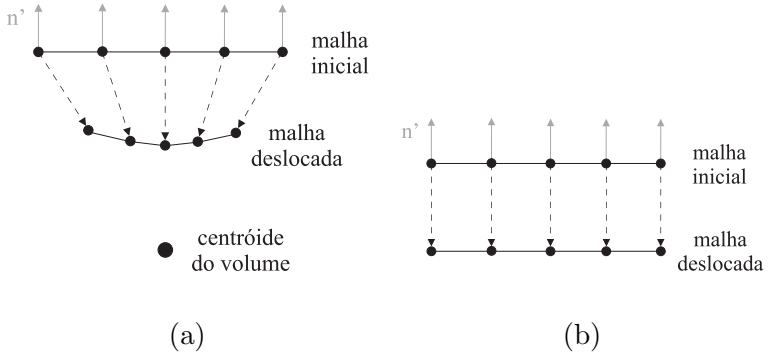


Figura 6: Estratégias de deslocamento dos vértices da malha. (a) deslocamento em direção ao centróide; (b) deslocamento na direção inversa do vetor normal.

de referência, a não ser que esta seja uma calota perfeita cujo centro é o centróide do volume. De fato, as malhas deslocadas não formam superfícies eqüidistantes entre si, como mostra a figura 6.

Nossa estratégia atual de deslocamento da superfície de referência consiste em estimar os vetores normais da malha suavizada utilizando o tradicional algoritmo de estimativa de normais com base na média das normais das faces adjacentes a cada vértice, e utilizar tais vértices para deteminar a direção de deslocamento.

A figura 8 mostra uma comparação do resultado da reformatação curvilinear usando essa estratégia de deslocamento e o deslocamento na direção do centróide, em um modelo visualizado com renderização volumétrica direta. No deslocamento em direção ao centróide, o recorte tem sempre o formato de uma cunha em direção ao centro do volume e não forma camadas eqüidistantes entre si (figura 8.b). No deslocamento utilizando o vetor normal da malha, a direção de recorte depende da curvatura da superfície e corresponde a camadas eqüidistantes (figura 8.c).

A malha triangular obtida a partir do desenho 3D sobre a isosuperfície da cabeça do paciente é geralmente convexa na direção dos vetores normais. Em nossa estratégia de deslocar os vértices na direção do vetor normal estimado, o deslocamento é realizado na

direção oposta dos vetores normais (para dentro da cabeça), de sorte que podem ocorrer auto-interseções. O problema de tratamento de auto-interseções de malhas triangulares de deslocamento é bem conhecido na literatura [13, 9]. Entretanto, por limitações de tempo optamos por tratar as auto-interseções através de um método que preserva a conectividade, evitando a inclusão de estruturas de dados complexas para armazenar a malha. Cada vez que a malha é deslocada, verificamos quais triângulos tiveram sua orientação invertida com relação aos triângulos da malha anterior. Tais triângulos invertidos correspondem a regiões de auto-interseção e devem ser removidos. Para evitar mudanças de topologia na malha, optamos simplesmente por reunir os vértices do triângulo invertido em um único ponto: seu centróide. Desse modo, o triângulo é degenerado em um ponto e seus triângulos adjacentes são degenerados em arestas. Mantemos uma estrutura de dados que conserva essas alterações da malha de modo que, em agrupamentos de vértices subseqüentes, grupos de vértices já agrupados se comportam como se fossem um único vértice e, assim, podem ser agrupados novamente com novos vértices.

3.5 Voxelização

Em nossa técnica, a voxelização se resume no processo de converter uma malha triangular a partir de sua representação geométrica contínua em um conjunto de *voxels* que melhor aproxime a malha. Esse conjunto de *voxels* correspondentes pode ser rotulado para remoção, possibilitando a visualização do interior do volume antes sob oclusão dos *voxels* rotulados. Na reformatação curvilinear, a iteração do processo de voxelização, rotulação e remoção se assemelha ao processo de "descascar uma cebola", camada por camada. A voxelização é o equivalente em 3D ao processo de varredura que rasteriza objetos geométricos 2D na tela. Certamente, em nosso caso a voxelização não é utilizada para desenhar *voxels*, mas meramente discretizar a malha contínua de modo a conhecer sua correspondência nos dados volumétricos.

Se tivermos um triângulo definido pelos pontos 3D $\mathbf{v}_1 = (x_1, y_1, z_1)$, $\mathbf{v}_2 = (x_2, y_2, z_2)$ e $\mathbf{v}_3 = (x_3, y_3, z_3)$, podemos expressar qualquer ponto $\mathbf{p}$ neste triângulo como uma combinação baricêntrica dos vértices $\mathbf{v}_1$, $\mathbf{v}_2$ e $\mathbf{v}_3$

$$\mathbf{p} = \alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3$$

com a restrição de que as coordenadas baricêntricas α, β e γ estejam no intervalo $[0, 1]$ e satisfaçam a equação

$$\gamma = 1 - \alpha - \beta.$$

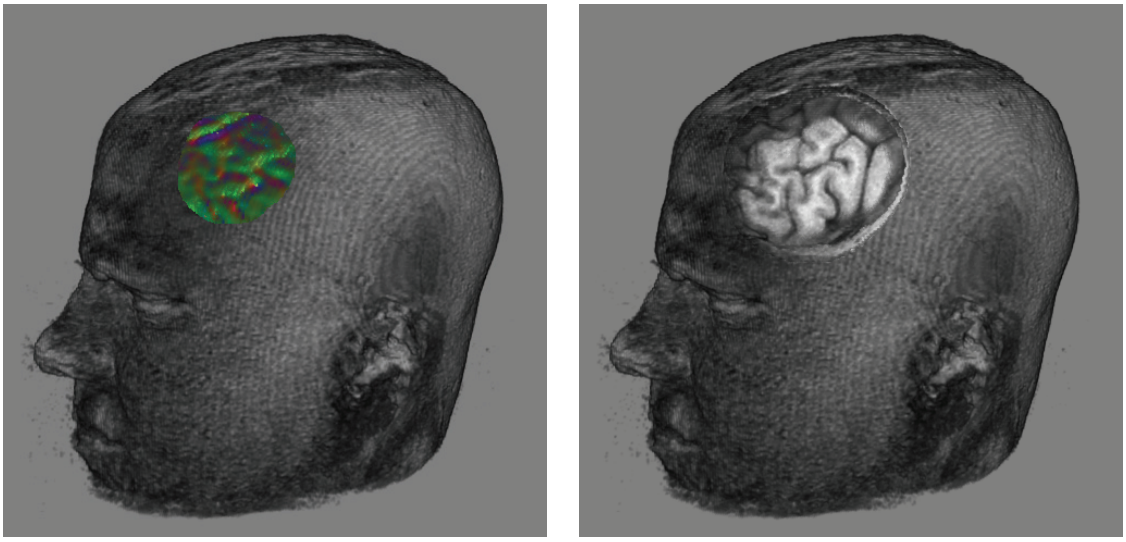
A princípio, podemos obter uma versão *raster* de um triângulo 3D através do rastreamento do intervalo $[0, 1]$ de modo a obter todas as possíveis combinações de coordenadas baricêntricas. Uma vez que as coordenadas de $\mathbf{p}$ estão geralmente restritas a valores inteiros, estimamos os passos mínimos de incremento $\Delta\alpha$ e $\Delta\beta$ por questões de eficiência:

$$\begin{aligned} \Delta\alpha &= \frac{1}{dist(\mathbf{p}, \overline{\mathbf{v}_2\mathbf{v}_3})} \\ \Delta\beta &= \frac{1}{dist(\mathbf{p}, \overline{\mathbf{v}_1\mathbf{v}_3})}, \end{aligned}$$

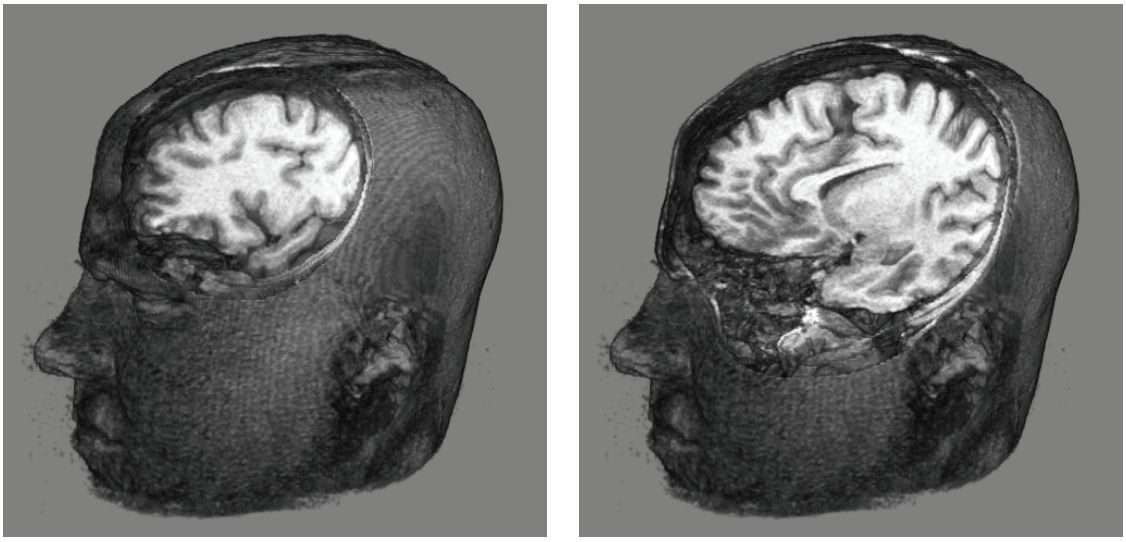
onde $dist(p, L)$ é a distância entre um ponto p e a linha L.

3.6 Recorte Irregular Usando a Transformada de Distância

Durante o desenvolvimento do protótipo, experimentamos o uso de uma transformada de distância Euclidiana 3D para remoção de camadas eqüidistantes de *voxels* com relação a uma superfície de referência. Nesse modo de interação, o usuário inicialmente desenha uma superfície de referência, a qual é voxelizada de modo a obter os *voxels* considerados como obstáculos na transformada, isto é, os *voxels* que serão utilizados como referência para calcular a distância entre todos os outros *voxels*. A transformada de distância é calculada na CPU e o resultado é armazenado na GPU em uma textura 3D contendo as distâncias. O usuário seleciona então um valor de distância de tal modo que apenas os *voxels* com distância maior ou igual a esse valor sejam levados em consideração na execução do algoritmo de *ray-casting*.



(a) (b)



(c) (d)

Figura 7: Recorte irregular usando a transformada de distância. (a) área rotulada; (b, c, d) remoção de *voxels* em diferentes distâncias da área rotulada.

Efetivamente, isso produz um efeito que lembra a reformatação curvilinear como a realizada com o deslocamento da malha, mas que é diferente desta porque utiliza apenas a distância entre os *voxels* e não a direção dos vetores normais da superfície de referência. De fato, para obter camadas eqüidistantes, a superfície de referência deve ser fechada. Para isso ocorrer, toda a superfície externa do cérebro precisa ser segmentada [5].

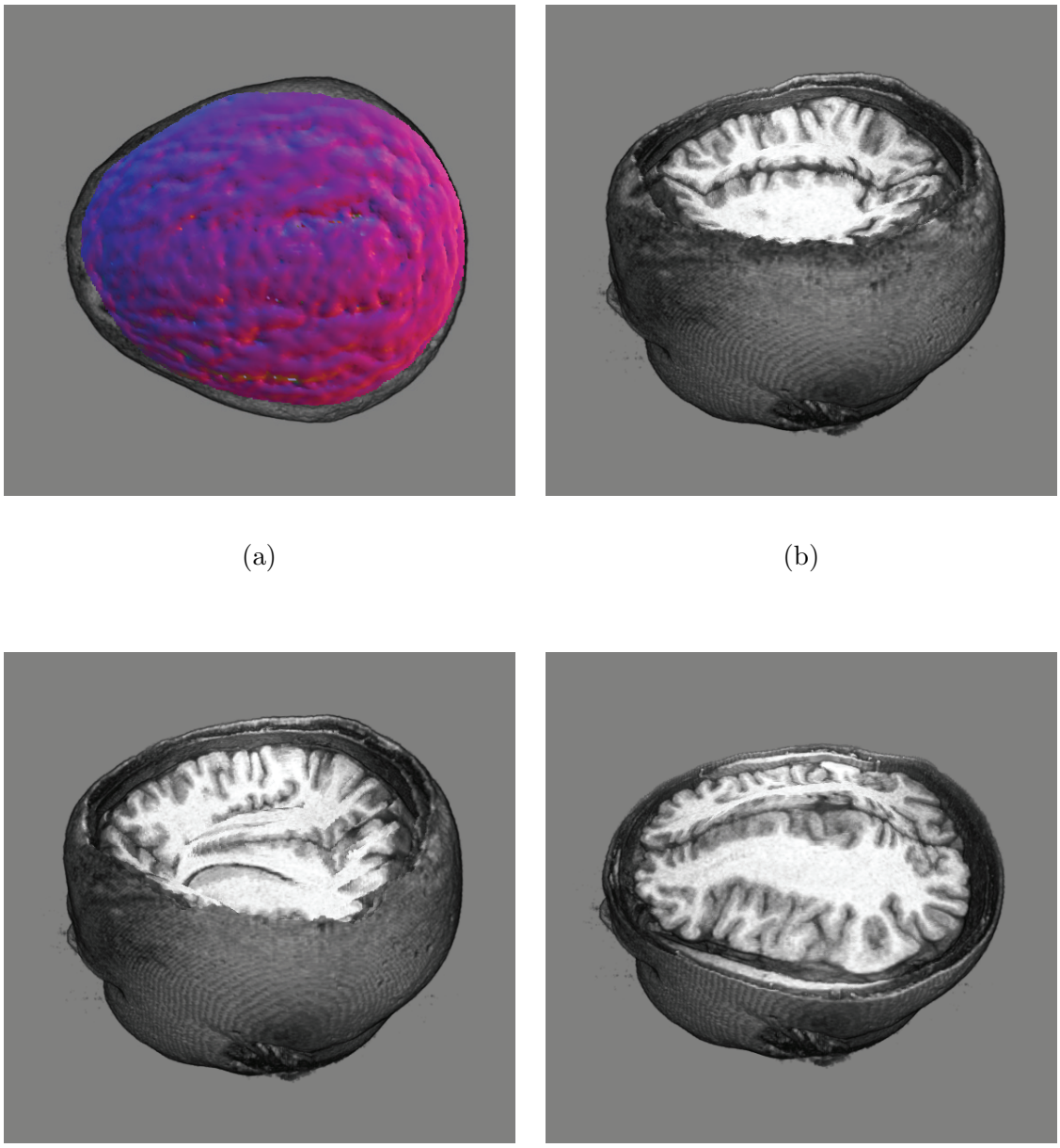
A figura 7 mostra o procedimento de desenho da superfície de referência e imagens do modelo após o cálculo da transformada de distância usando diferentes valores de distância para realizar o recorte. A figura 8 compara o recorte irregular realizado com o deslocamento da malha triangular e a transformada de distância, dado uma mesma superfície de referência e profundidade (distância, no caso da transformada de distância) de recorte de 60 *voxels*.

4 Discussão e Resultados

Um protótipo de nossa ferramenta de reformatação curvilinear foi implementado com base na integração da arquitetura de *ray-casting* na GPU de Stegmaier *et al.* [19] e nossa arquitetura de interação [2] ajustada ao uso de dados volumétricos de acordo com os procedimentos apresentados na seção 2. Os algoritmos foram implementados em OpenGL/C++ com *shaders* em *assembly* do OpenGL. A API wxWidgets foi utilizada inicialmente para implementar parte da interface 2D. Todas essas bibliotecas e APIs são livres.

O protótipo foi avaliado em um computador Intel Core2 Duo E8500 3.16Hz com 2GB RAM, e uma placa gráfica NVIDIA GeForce 9600GT com 512MB RAM. Testamos a reformatação curvilinear em uma série de dados cerebrais obtidos do *scanner* Elscint Prestige 2T do Hospital das Clínicas da Unicamp. A resolução dos dados foi de cerca de $256x256x180$ *voxels*, compostos de uma séria de imagens do plano coronal adquiridas em fatias de 1mm de espessura.

Na inicialização, a aplicação exibe duas janelas: uma com a visualização de isosuperfícies do volume e outra com a visualização utilizando a técnica de renderização volumétrica direta,



(a) (b)

(c) (d)

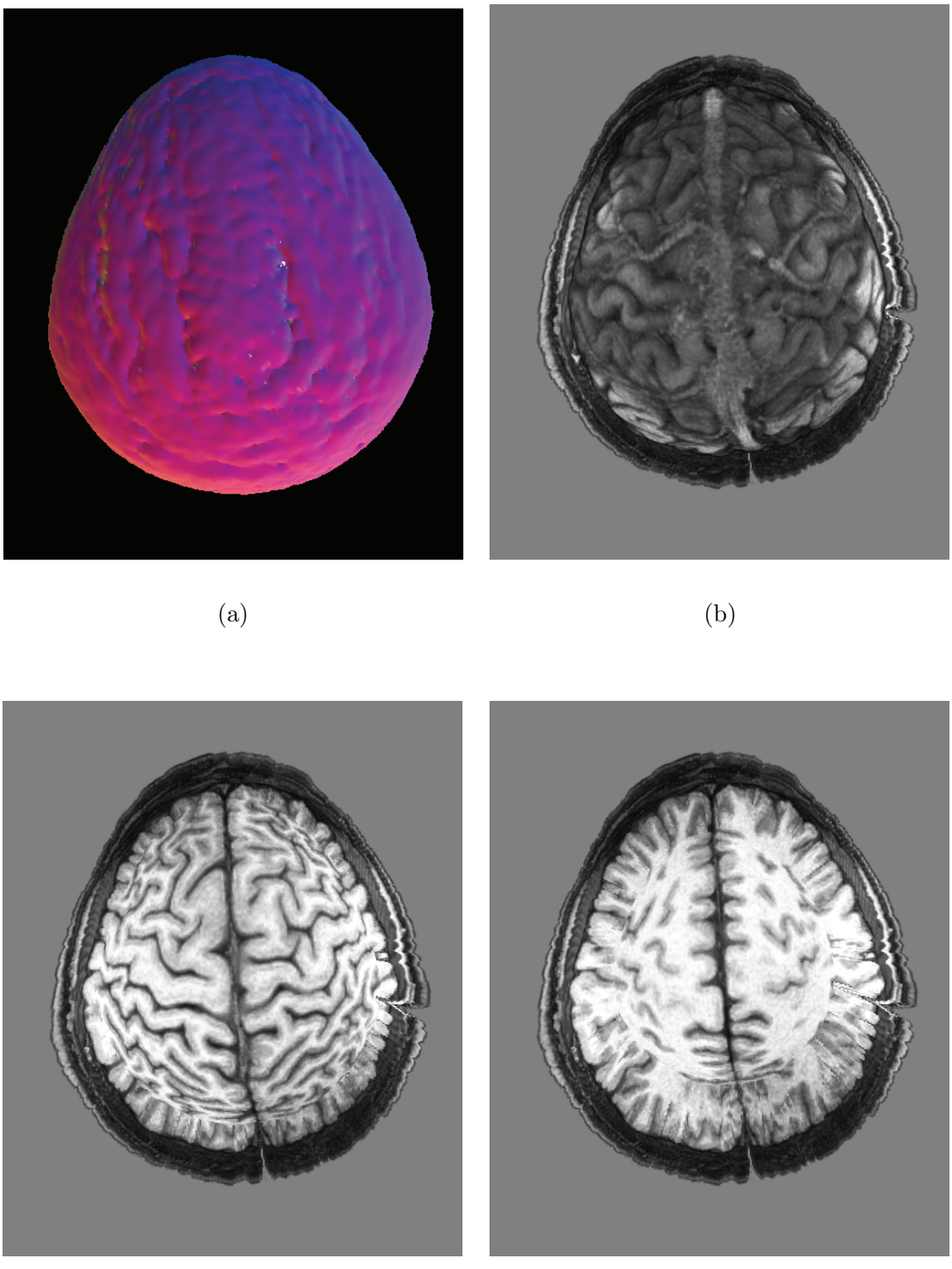
Figura 8: Comparação entre recorte irregular usando deslocamento da malha triangular e transformada de distância. (a) superfície de referência; (b) deslocamento na direção do centróide do volume; (c) deslocamento na direção inversa do vetor estimado em cada vértice; (d) usando transformada de distância.

como mostra a figura 1. O usuário pode utilizar o dispositivo apontador 2D para guiar um cursor 3D sobre as isosuperfícies visualizadas. Tal cursor é o mesmo cursor tríade utilizado na arquitetura de interação do trabalho de doutorado, e tem a habilidade de se fixar na isosuperfície apontada pelo dispositivo apontador 2D, ajustando a representação visual do vetor normal de acordo com o vetor normal obtido do ponto de interesse. Isso dá ao usuário a sensação de estar deslizando diretamente sobre o escalpo.

Assim como em uma aplicação de desenho de imagens, o usuário pode pintar áreas das isosuperfícies utilizando um pincel virtual. As áreas pintadas indicam as áreas correspondentes à superfície de referência da reformatação, isto é, onde o volume será "descascado." Para a reformatação curvilinear, o usuário seleciona um isovalor que corresponde à visualização de isosuperfícies da pele da cabeça do paciente. A pintura é feita então sobre tal isosuperfície. Após a pintura, o usuário seleciona a profundidade da camada a ser descascada e a reformatação curvilinear é então aplicada em tempo real. A visualização nas duas janelas é atualizada e o usuário pode guiar o cursor 3D sobre o volume descascado e as novas isosuperfícies visualizadas (usando uma tecla de atalho para selecionar o nível de profundidade desejado). A figura 9 mostra visualizações da reformatação curvilinear dos dados do cérebro em diferentes profundidades.

A eficiência depende principalmente do desempenho do algoritmo de *ray-casting* na GPU. Porém, em todos os casos testados, os dados volumétricos puderam ser visualizados e manipulados em tempo real. Isso difere significativamente de outras ferramentas de reformatação curvilinear. Por exemplo, a técnica de reformatação curvilinear automática de Huppertz *et al.* [8] não opera em tempo real. Em particular, requer entre 1 a 1,5 minutos de processamento para realizar a etapa de normalização dos dados de origem para o modelo padrão utilizado pelo *software* SPM2 (*Statistical Parametric Mapping*) [6].

Em nosso protótipo também implementamos outras formas de recorte comuns em sistemas de visualização de dados volumétricos, tais como o recorte por um plano orientado



(a) (b)

(c) (d)

Figura 9: Dados volumétricos com camadas removidas em diferentes profundidades: (a) área pintada; (b) 15 *voxels* removidos em profundidade a partir da superfície de referência; (c) 30 *voxels* removidos; (d) 50 *voxels* removidos.

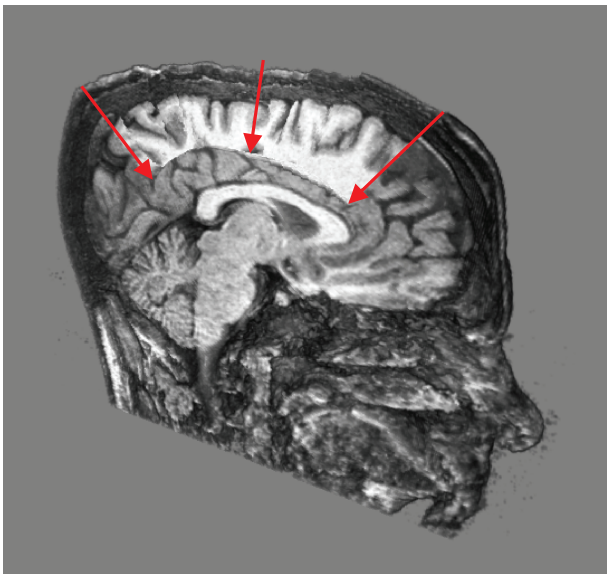


Figura 10: Visualização com renderização volumétrica direta utilizando combinação de reformatação curvilinear e plano de recorte sagital. A profundidade e direção do corte curvilinear é indicada pelas setas.

livremente. A figura 10 mostra o modelo da cabeça de um paciente visualizado com a aplicação de um plano de recorte sagital, combinado com a reformatação curvilinear.

5 Conclusões

Neste trabalho exploramos o uso da arquitetura de interação proposta no doutorado de modo a realizar interação com dados volumétricos obtidos a partir de imagens médicas de ressonância magnética. Nossa meta é prover funcionalidades de interação que estreitem a lacuna existente entre o que o usuário vê e aquilo com que o usuário interage. Isso é proposto com base na tese de que propriedades geométricas locais são suficientes para a realização de uma interação acurada, portanto podemos realizar manipulações diretas utilizando modelos cujas representações existem apenas na memória de vídeo. Tal é o caso dos dados volumétricos por nós utilizados. Utilizamos como base a arquitetura de visualização volumétrica de Stegmaier *et al.* [19], que armazena dados volumétricos como texturas 3D e explora o *hardware* gráfico programável para visualizar tais dados através de *ray-casting*. Em nossa arquitetura de interação, esse processamento se concentra no processador de fragmentos, de forma muito semelhante a algoritmos de mapeamento de detalhes 3D baseados em *ray-casting*, como a técnica de *relief mapping* [14]. Pressupondo que a interação pode ser feita com base nas isosuperfícies detectadas por *ray-casting*, podemos configurar facilmente nossa arquitetura de modo a interagir com dados volumétricos. De modo a validar essa possibilidade, desenvolvemos uma ferramenta de reformatação curvilinear e recorte irregular de dados volumétricos, utilizando uma metáfora de pintura 3D para realizar a segmentação que resulte na superfície de referência da reformatação. Essa abordagem foi realizada utilizando como premissa a suficiência da interação com isosuperfícies para a realização dos cálculos de atributos geométricos.

Vídeos sobre a ferramenta de reformatação curvilinear e recorte irregular estão disponíveis na página da internet: `http://www.dca.fee.unicamp.br/projects/mtk/batageloP`.

A idéia chave de nossa proposta é integrar uma ferramenta eficiente de visualização de imagens 3D de ressonância magnética, a qual pode produzir imagens de alta qualidade de acordo com o foco de atenção do usuário, e utilizar um dispositivo apontador 2D para permitir a localização precisa de um cursor 3D sobre um *voxel* visível de interesse. No caso da segmentação 3D, os *voxels* de interesse são *voxels* que servem de interface de duas regiões perceptualmente discerníveis por um especialista e facilmente marcadas com a ajuda de um dispositivo apontador 2D usando nossa arquitetura de interação. Alguns ajustes foram necessários de modo a acomodar as disparidades entre os dados contínuos e discretos. Em especial, utilizamos operações com malhas triangulares de modo a definir a superfície de referência da reformatação curvilinear a partir dos *pixels* pintados na tela, e operações de voxelização para realizar a correspondência entre a superfície de referência e os *voxels* a serem rotulados para remoção.

Como a arquitetura de interação é baseada apenas nas informações disponíveis no espaço da imagem, lembramos que, se o cérebro inteiro precisa ser segmentado, pelo menos dois pontos de vista são necessários. Segundo nossa abordagem, isso significa que pelo menos duas malhas triangulares terão de ser geradas separadamente e costuradas de modo a formar um único envelope. Atualmente nossa ferramenta trabalha apenas com a malha triangular obtida de um único ponto de vista. Desse modo, tal processo é um dos pontos de investigação futura deste trabalho. Outra suposição que tivemos em nossos testes experimentais era que a renderização volumétrica direta e visualização de isosuperfícies gerariam resultados visuais similares. Na prática, verificamos que não. Uma interação apenas com isosuperfícies não é suficiente para prover uma realimentação dentro da expectativa do usuário que permita, por exemplo, caminhar com um cursor 3D sobre o cérebro, pois as pequenas diferenças de densidades dos tecidos no interior do cérebro geram isosuperfícies que são percebidas erroneamente como ruído. Idealmente, a superfície de interação deve corresponder ao que o usuário está de fato visualizando, o que depende do modo de visualização. Em nosso caso

constatamos que a renderização volumétrica direta é a preferida, de modo que a visualização depende intimamente da função de transferência utilizada. Entretanto, internamente a interação é realizada no momento apenas sobre isosuperfícies. Acreditamos que uma interação mais coerente é possível se levarmos em consideração superfícies calculadas de acordo com o resultado aparente do volume visualizado.

Não obtivemos publicações no período de realização deste trabalho. Isso se deve ao estado incipiente do protótipo atual, que ainda não permite seu uso efetivo em um ambiente clínico. O emprego da ferramenta em ambiente clínico, de modo a validar os aspectos de facilidade de uso e eficiência que ressaltamos, é essencial segundo os comentários obtidos de revisores de dois congressos para os quais enviamos nossos artigos: o EuroVIS (*Eurographics/IEEE Symposium on Visualization*) 2009 e ISBI (*IEEE International Symposium on Biomedical Imaging*) 2009. A adequação do protótipo para uso em ambiente clínico é, portanto, nossa prioridade para trabalhos futuros a curto prazo.

6 Trabalhos Futuros

Dado o interesse da comunidade médica do LNI em utilizar nossas ferramentas de interação, e interesse em vincular esse projeto ao programa CInAPCe (Cooperação Interinstitucional de Apoio a Pesquisas sobre o Cérebro) da FAPESP, pretendemos dar continuidade a este projeto, sugerindo transformar o protótipo atual em um produto que possa ser utilizado em ambiente clínico. Tal produto incluirá a técnica de reformatação curvilinear proposta neste trabalho, além das tradicionais técnicas de recorte regular e ferramentas de medição de ângulos e distâncias. Para atingir esse objetivo, propomos os seguintes melhoramentos sobre o protótipo atual a curto prazo para que os conceitos implementados possam ser validados:

- Possibilitar a pintura de malhas de referência em diferentes pontos de vista, realizando a costura automática dessas malhas de modo a formar um único envelope.

- Utilizar estruturas de dados robustas para calcular o deslocamento das malhas de referência e resolver possíveis auto-interseções.

Além desses objetivos a curto prazo, pretendemos estudar dois problemas de pesquisa ainda em aberto. O primeiro consiste em integrar, na arquitetura de visualização através de *ray-casting*, a visualização dos *widgets* 3D, tais como o cursor 3D e as marcações de ângulos e distâncias. Atualmente, a visualização desses *widgets* é realizada em um passo adicional de renderização usando malhas poligonais, não respeitando assim o efeito de translucência do modelo volumétrico visualizado pelo método de renderização volumétrica direta. O segundo problema consiste em realizar a interação a partir de superfícies calculadas de acordo com parâmetros que levem em consideração a função de transferência utilizada na visualização. Até este momento, consideramos apenas o uso de isosuperfícies, conseguindo resultados satisfatórios para a ferramenta de reformatação curvilinear. Ao levar em consideração o processamento que decorre do uso de funções de transferência e avaliação da equação de renderização, poderemos integrar em nosso protótipo ferramentas adicionais de interação, tais como delineamento de sulcos e giros cerebrais utilizando o cursor 3D, e fornecer uma realimentação visual mais coerente com aquilo que o usuário percebe visualmente.

6.1 Cronograma

A realização dos trabalhos futuros seguirá o seguinte cronograma compreendido em cinco etapas dentro do período de 12 meses:

- Etapa 1 (mês 1): Conversão do código do protótipo atual de modo a utilizar *shaders* GLSL (OpenGL 3.0) em vez dos *shaders* em *assembly* do OpenGL derivados da implementação de Stegmaier *et al.*
- Etapa 2 (mês 2 a 4): Implementação dos melhoramentos a curto prazo para utilização do protótipo atual em ambiente clínico.

- Etapa 3 (mês 5 a 6): Análise das soluções do problema de integração da renderização volumétrica por *ray-casting* e renderização de *widgets* 3D, e implementação no protótipo atual.
- Etapa 4 (mês 7 a 8): Estudo do problema de interação com superfícies definidas segundo parâmetros que melhor aproximem a percepção visual do usuário no modo de renderização volumétrica direta.
- Etapa 5 (mês 9 a 12): Utilização das soluções obtidas na etapa anterior para implementação de melhoramentos adicionais do protótipo (*e.g.*, ferramentas de suporte a planejamento cirúrgico), resultando em um produto.

Ao término das etapas 2, 4 e 5 pretendemos publicar um artigo contendo os resultados do uso do protótipo com as funcionalidades implementadas.

Referências

[1] Alexandre C. Bastos, Roch M. Comeau, Frederick Andermann, Denis Melanson, Fernando Cendes, Francois Dubeau, Suzanne Fontaine, Donatella Tampieri, and Andre Olivier. Diagnosis of subtle focal dysplastic lesions: Curvilinear reformatting from three-dimensional magnetic resonance imaging. *Annals of Neurology*, 46(1):88–94, 1999.

[2] Harlen Costa Batagelo and Shin-Ting Wu. A framework for GPU-based application-independent 3d interactions. *The Visual Computer*, 24(12):1003–1012, 2008.

[3] F.P.G. Bergo and A. X. Falcão. Fast and automatic curvilinear reformatting of MR images of the brain for diagnosis of dysplastic lesions. In *2006 International Symposium on Biomedical Imaging*, pages 486–489, Arlington, Virginia, USA, 2006.

[4] L. P. Clarke, R. P. Velthuizen, M. A. Camacho, J. J. Heine, M. Vaidyanathan, L. O. Hall, R. W. Thatcher, and M. L. Silbiger. MRI segmentation: Methods and applications. *Magnetic Resonance Imaging*, 13(3):343–368, 1995.

[5] Klaus Engel, Markus Hadwiger, Joe Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-Time Volume Graphics*. A K Peters Ltd., first edition, 2006.

[6] Guillaume Flandin and Karl J. Friston. SPM2 - Statistical Parametric Mapping. `http://www.fil.ion.ucl.ac.uk/spm/software/spm2/`, 2009.

[7] Solomon W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.

[8] H.-J. Huppertz, J. Kassubekb, D.-M. Altenmüllerc, T. Breyerd, and S. Fauserc. Automatic curvilinear reformatting of three-dimensional mri data of the cerebral cortex. *NeuroImage*, 39(1):80–86, January 2008.

[9] Wonhyung Jung, Hayong Shin, and Byoung K. Choi. Self-intersection removal in triangular mesh offsetting. *Computer-Aided Design & Applications*, 1(1–4):477–484, 2004.

[10] Mathieu Malaterre. *GDCM Reference Manual*. `http://gdcm.sourceforge.net/gdcm.pdf`, first edition, 2008.

[11] Marcelo De Gomensoro Malheiros, Flavio Navarro Fernandes, and Shin-Ting Wu. MTK: A direct 3D manipulation toolkit. In *Proceedings of SCCG '98*, pages 81–88, April 1998.

[12] Herman Oosterwijk. DICOM reference guide. *Health Devices*, 30(1-2):5–30, January 2001.

[13] Sang C. Park. Triangular mesh intersection. *The Visual Computer*, 20(7):448–456, September 2004.

[14] Fábio Policarpo, Manuel M. Oliveira, and João L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pages 155–162, Washington, DC, USA, April 2005. ACM Press.

[15] Christof Rezk-Salama, Klaus Engel, and Fernando Vega Higuera. The OpenQVis project. `http://openqvis.sourceforge.net/`, 2009.

[16] Robert F. Rice and J. R. Plaunt. Adaptive variable-length coding for efficient compression of spacecraft television data. *IEEE Transactions on Communications*, 16(9):889–897, December 1971.

[17] Rogue Research Inc. *Brainsight - User Manual, Version 1.7*, 2001.

[18] Simon Stegmaier, Magnus Strengert, and Thomas Klein. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. `http://www.vis.uni-stuttgart.de/eng/research/fields/current/spvolren/`.

[19] Simon Stegmaier, Magnus Strengert, Thomas Klein, and Thomas Ertl. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Proceedings of Volume Graphics 2005*, pages 187–195, Stony Brook, New York, USA, June 2005.

[20] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 329–346, 1963.

[21] Filippo Tampieri. Newell's method for computing the plane equation of a polygon. In *Graphics Gems III*, pages 231–232. Academic Press Professional, Inc., San Diego, CA, USA, 1992.

[22] Colin Ware. *Information Visualization: Perception for Design.* Morgan Kaufmann Publishers, second edition, 2004.

[23] Shin-Ting Wu, Marcelo Abrantes, Daniel Tost, and Harlen Costa Batagelo. Picking and snapping for 3d input devices. In *XVI Brazilian Symposium on Computer Graphics and Image Processing*, pages 140–147, October 2003.